A Semantic Web-based Infrastructure Supporting Context-Aware Applications

Renato F. Bulcão Neto†, Cesar A. C. Teixeira‡, and Maria da Graça C. Pimentel†

†Universidade de São Paulo, São Carlos-SP, 13560-970, Brazil
‡Universidade Federal de São Carlos, São Carlos-SP, 13560-970, Brazil
rbulcao@icmc.usp.br, cesar@dc.ufscar.br, mgp@icmc.usp.br

Abstract. There is a demand for efforts that deal with the challenges associated to the complex and time-consuming task of developing context-aware applications. These challenges include context modeling, reuse and reasoning, and software infrastructures intended to context management. This paper presents a service infrastructure for the management of semantic context called Semantic Context Kernel. The novelty is a set of semantic services that can be personalized according to context-aware applications' requirements so as to support the prototyping of such applications. The Semantic Context Kernel has been built upon an ontological context model, which provides Semantic Web abstractions to foster context reuse and reasoning.

1 Introduction

One of the research themes in ubiquitous computing[1] is the context-aware computing, where applications customize their behavior based on context information sensed from those instrumented environments. A classic definition of context [1] is "any relevant information about the user-application interaction, including the user and the application themselves". For instance, by means of sensors network and computers, the PROACT elder care system infers whether and how people with early-stage cognitive decline perform activities of daily living [2].

There is a demand for efforts that deal with the challenges associated to the complex and time-consuming task of developing context-aware applications [3]. Challenges related to the development of such applications include: (i) how to represent context in such manner that facilitates its sharing, reuse and processing; (ii) the development of software infrastructures to support applications in respect with context management. High-level context models need representation languages that use broadly accepted standards so as to facilitate the sharing and the reuse of context [4]. Moreover, the more formal a context model is, the better is the ability for context-aware applications to reason about context. A software infrastructure built on top of such context models can then provide context-aware applications with enhanced-services intended to exploit context sharing, reuse and reasoning.

[1] The term "ubiquitous computing" is hereafter referred to "ubicomp".

The literature has reported that the Semantic Web vision [5] fits well the need for context models that enables applications to process the semantics of context even regardless applications domains. The GaiaOS middleware is able to reason about context represented as first-order predicates [6]. The CoBrA agent architecture acquires, manages and reasons about shared ontological context, and also detects and resolves inconsistent context [4]. The Semantic Spaces infrastructure supports the inference of higher-level contexts from basic contexts in which the semantics of context is also explicitly conveyed by ontologies [7]. Although those efforts address important issues such as context classification and dependency, and quality and privacy of context, none have focused on providing applications with semantic services that can be configured according to applications' requirements.

This paper presents a Semantic Web-based service infrastructure for context management called *Semantic Context Kernel*. Its architecture is composed of configurable semantic services for context storage, query and reasoning, and service discovery. The Semantic Context Kernel has been built upon an ontological context model [8] that provides a general vocabulary so lower ontologies can import it for particular domains. The design space for building this context model derives from dimensions for context modeling debated in the literature: identity (*who*), location (*where*), time (*when*), activity (*what*) [9] and devices (*how*) [10]. In order to address these five context dimensions, concepts of well-known semantic web ontologies have been reused and extended.

The novelty regarding the Semantic Context Kernel relies on its configurability feature: services can be personalized according to context-aware applications' requirements so as to facilitate the prototyping of such applications. For instance, the context persistence service allows application developers to choose the type of persistent storage (e.g. files and databases) and the type of serialization [11] (e.g. RDF/XML and N-Triples) for semantic context. Different levels of inference over context can be exploited by means of the context inference service (e.g. transitive and rules-based reasoning).

Section 2 outlines Semantic Web standards that we have exploited to build our context model. In Section 3 we present our ontology-based context model. Section 4 describes the architecture of the Semantic Context Kernel. Section 5 illustrates the use of that infrastructure by an application on the educational domain. Finally, in Section 6 we present concluding remarks and future work.

2 Semantic Web Background

In order to exploit the full potential of the Semantic Web, there is a need for standards for describing resources — anything that can be uniquely identified — in a language that makes their meaning explicit. In order to explicitly associate meaning for data, knowledge must be represented in some way. One attempt to apply ideas from knowledge representation is the RDF standard [12].

The RDF specification provides a generic data model that consists of nodes connected by labeled arcs: nodes represent resources, and arcs represent properties or relations used to describe resources. A resource together with a property and the value of that property for that resource is called an RDF statement. Those three individual parts of a statement form the RDF triple model. The RDF Schema language [13] conveys the semantics of RDF metadata by means of mechanisms for describing classes of resources, relationships between resources, and restrictions on properties.

The OWL language [14] is a step further for the creation of controlled, shareable, and extensible vocabularies. OWL builds on RDF and RDF Schema and adds more vocabulary for describing properties and classes, including among others: relations between classes (e.g. disjointness, inverse), (in)equality between individuals, cardinality (e.g. exactly one, at least one), richer typing of properties (e.g. enumerated datatypes), characteristics of properties (e.g. symmetry, transitivity, functional), and constraints on properties (e.g. all values from, some values from, cardinality).

The next section describes our domain-independent ontological context model. It is based on the OWL ontology language due to the following reasons: (i) it provides formal semantics for reasoning about context; (ii) it is compatible with several standard specifications intended to the Semantic Web; (iii) it allows ontologies to be distributed across many systems; and (iv) its openness and extensibility.

3 An Ontological Context Model for UbiComp Applications

Figure 1 depicts our ontological context model described elsewhere [8]. It represents the basic concepts of actors, location, time, activities, and devices as well as the relations between these concepts. Concepts of semantic web vocabularies have been borrowed in order to address every dimension as well as to serve as guidance for context modeling. We describe our context model as follows.

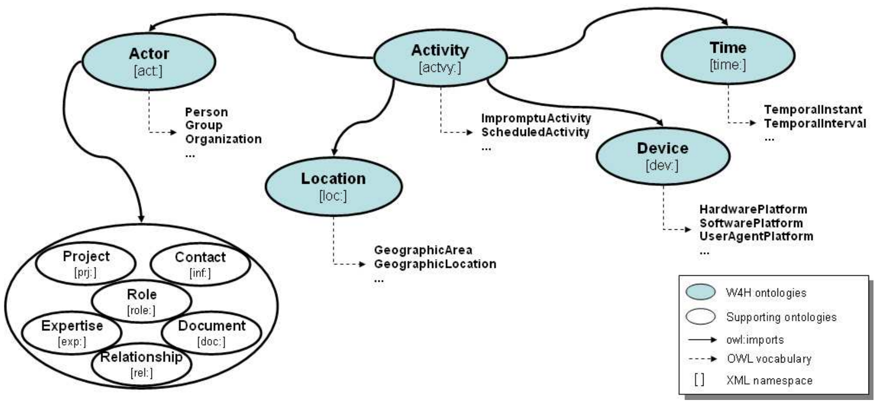

Fig. 1. A domain-independent semantic context model with high-level ontologies.

In **Figure** 1, the *Actor* ontology models the profile of entities performing actions in a ubiquitous computing environment such as people, groups and organizations. This ontology imports other ontologies that we have built to deal with actors' profile: *knowledge*, *social relationship*, *document*, *social role*, *contact information* and *project*. The knowledge ontology models knowledge areas so as to relate someone to a particular expertise or topic of interest. The relationship ontology describes people's social network (e.g. *cooperatesWith*). The document ontology models documents made by actors such as web pages. The role ontology describes the actors' social role in the real world (e.g. *student*). The contact ontology represents different types of actors' contact information (e.g. *email*). Finally, the project ontology models meta-information associated to projects and actors (e.g. *isHeadedBy*).

The *Location* ontology describes the whereabouts of actors. It models indoor and outdoor places (e.g. *room* and *parking lot*), containment and spatial relations between places (e.g. *isPartOf* and *isConnectedTo*), and geographic coordinates (e.g. *latitude* and *longitude*). This ontology also represents places with respect to the address that they are located (e.g. *zip code*). In other words, a building can be related to the address where it is located (e.g. *street* and *zip code*).

The *Time* ontology represents time in terms of temporal instants and intervals [15]. We modeled temporal relations between instants and intervals (e.g. an instant is *insideOf* an interval), properties of intervals (e.g. the *durationOf*), and temporal relations between intervals (e.g. *equals*, *starts* and *finishes*). The temporal ontology also provides a standard way of representing calendar and clock information on the Semantic Web.

The *Device* ontology describes devices features regarding its *hardware*, *software* and *user agent* platforms. The hardware platform describes the I/O and network features of a device (e.g. whether a display is color-capable). The software platform describes application environment, operating system, and installed software (e.g. the types of Java virtual machines supported). The user agent platform describes the web browser running on a device (e.g. whether it is Javascript-enabled).

The *Activity* ontology describes actions that actors do or cause to happen. We modeled an activity as a set of relevant events that characterizes it. An event is a fact that includes *spatiotemporal* descriptions, as well as descriptions of the corresponding *actors* and *devices* involved. The relevance, the type and the combination of events for inferring activities are dependent on the user's task in the current domain.

The next section presents the service infrastructure that we have built for the management of semantic context.

4 The Semantic Context Kernel

Built upon our semantic context model, we have implemented a service infrastructure called *Semantic Context Kernel*. The aim is to provide developers with a set of semantic-enabled services that can be configured so as to address applications' requirements. Since this work handles the semantics of context, it furthers our previous work on software infrastructure for context awareness [16]. **Figure** 2 depicts the Semantic Context Kernel architecture.

In **Figure** 2, context information is provided by *context sources*, which include applications, web services, and physical sensors. *Context transducers* convert the information captured from context sources into a common semantic representation: the RDF triple model. This approach addresses both interoperability and reuse issues. *Context consumers* make use of context information stored by context sources so that the former can adapt themselves following the current situation (e.g. applications). The *dis-*

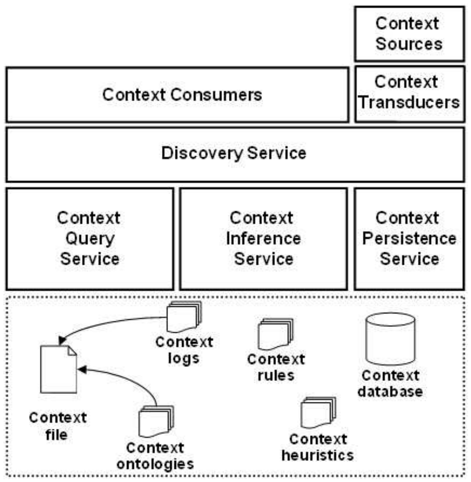

Fig. 2. The Semantic Context Kernel architecture.

covery service provides context transducers and every service layer with an advertising mechanism so as to allow context consumers to locate these services.

The *context query service* allows context consumers to query context through a declarative language for RDF models that support simple conjunctive triple patterns called RDQL (*RDF Data Query Language*) [17]. In the general case, query expressions are represented as a matching of a triple pattern against an input source RDF graph. Example 1 describes an RDF query declared by a context consumer so as to obtain all sequence of triples matching the following constraint: the list of names (variable *?name*) and corresponding chat IDs of type ICQ (variable *?icqValue*) of all people with some resource whose name is “Steve Orr” works. The result of the current query is as follows: “Ian Battle” and “10043355”.

```
<!--            Example 1            -->
SELECT ?name, ?icqValue
FROM   <file:sck/contextFile.nt>
WHERE  (?x <act:hasName> "Steve Orr")
       (?x <rel:worksWith> ?y)
       (?y <act:hasName> ?name)
       (?y <act:hasContactProfile> ?z)
       (?z <inf:imType> "ICQ")
       (?z <inf:imValue> ?icqValue)
USING  act for <http://linkserver/2005/actor.owl#>
       rel for <http://linkserver/2005/relationship.owl#>
       inf for <http://linkserver/2005/contactinfo.owl#>
```

The input source is a file (*FROM* clause) containing an RDF graph with all triples (see **Figure** 3) representing context information stored by context sources. In this case, the file *contextFile.nt* stores RDF triples using an alternative serialization called N-Triples, where each RDF triple is represented in a line-based, plain text format. Example 2 is the content of the context repository represented in N-Triples, which is a very suitable serialization for large RDF models.

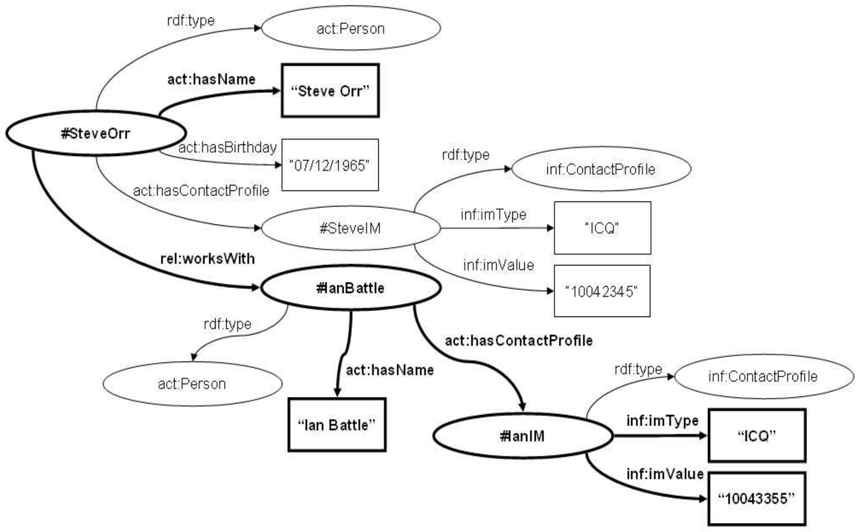

Fig. 3. This RDF graph represents the content of the context file identified by the FROM clause in the RDF query described in Example 1. The triples described in that query are printed in bold.

```
<!--            Example 2             -->
@prefix : <http://linkserver/example#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix act: <http://linkserver/2005/actor.owl#>.
@prefix rel: <http://linkserver/2005/relationship.owl#>.
@prefix inf: <http://linkserver/2005/contactinfo.owl#>.
:SteveOrr rdf:type act:Person.
:SteveOrr act:hasName "Steve Orr".
:SteveOrr act:hasBirthday "07/12/1965".
:SteveOrr act:hasContactProfile :SteveIM.
:SteveOrr rel:worksWith :IanBattle.
:SteveIM  rdf:type inf:ContactProfile.
:SteveIM  inf:imType "ICQ".
:SteveIM  inf:imValue "10042345".
:IanBattle rdf:type act:Person.
:IanBattle act:hasName "Ian Battle".
:IanBattle act:hasContactProfile :IanIM.
:IanIM  rdf:type inf:ContactProfile.
:IanIM  inf:imType "ICQ".
:IanIM  inf:imValue "10043355".
```

The *context inference service* provides context consumers with a configurable inference support over context. Developers can specify the level of inference to be supported over context, e.g. a transitive reasoner basically infers the hierarchical relations between classes (via *rdfs:subClassOf*). Example 3 shows some results of an inference process using a transitive reasoner over the RDF graph depicted in **Figure** 4.

```
<!--            Example 3             -->
Resource SteveOrr is instance of the class act:Person;
                                           act:Actor;
Resource SteveLocation is instance of the class loc:MeetingRoom;
                                                loc:Room;
                                                loc:IndoorLocation;
```

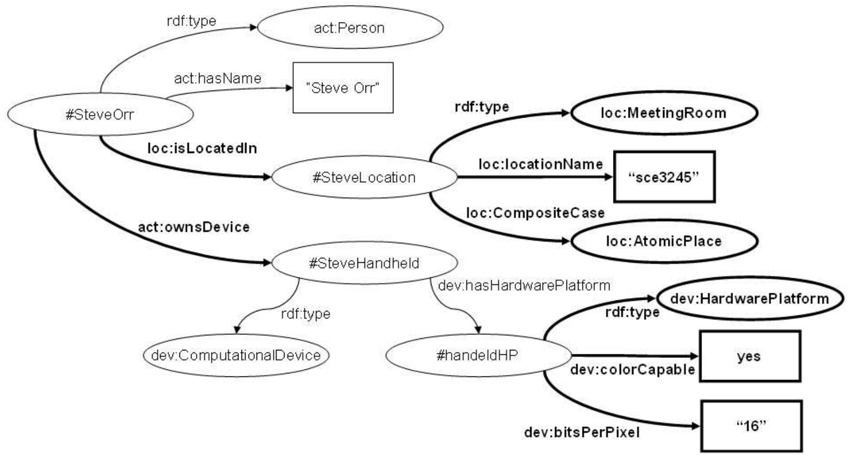

Fig. 4. This RDF graph describes a person called Steve Orr, located in a meeting room called sce3245, with a handheld device with color capability. Context information printed in bold depicts the relation between actors, locations and devices, and the characteristics of locations and devices.

On the other hand, in order to exploit high-level context, developers can define rules and store them on files, called *context rules*. When the context inference service is set up to use rules, it reads ontology facts into memory represented as RDF triples and parse those rules so as to validate them. When using rules, the context inference service allows developers to choose the type of reasoning to be performed, e.g. inductive, deductive and inductive-deductive reasoning. Example 4 presents two forward-chaining rules: the former describes that if a graduate student and his supervisor are in the meeting room, then they are attending a meeting; the latter describes that if two graduate students are members of a same study group and both are in the study room using the same tablet PC, then they are attending a study group meeting.

```
<!--            Example 4            -->
Person(A) ˆ hasRole(A,B) ˆ Person(C) ˆ hasRole(C,D) ˆ Graduate(B) ˆ Faculty(D) ˆ
isSupervisorOf(D,B) ˆ isLocatedIn(A,E) ˆ isLocatedIn(C,E) ˆ MeetingRoom(E)
--> attendingMeeting(A,C)

Person(A) ˆ hasRole(A,B) ˆ Person(C) ˆ hasRole(C,D) ˆ Graduate(B) ˆ Graduate(D) ˆ
isLocatedIn(A,E) ˆ isLocatedIn(C,E) ˆ StudyRoom(E) ˆ isMemberOf(A,F) ˆ
isMemberOf(C,F) ˆ StudyGroup(F) ˆ ownsDevice(A,G) ˆ ownsDevice(C,G) ˆ TabletPC(G)
--> StudyGroupMeeting(A,C)
```

If the inference process over context originates a conflict, user-defined *context heuristics* can be used to resolve it. An example of conflict can arise when a person is located in two different atomic places at the same time, e.g. a meeting room and a classroom with no relation of composition between them. This can be originated due to problems with freshness and accuracy of location information gathered from physical sensors.

The *context persistence service* allows developers to choose the types of persistent storage and context serialization. We allow to store context in relational databases as well as on a *context file*. The former approach can handle context in the RDF/XML and the N-Triples syntaxes, whereas the latter represents context in N-Triples.

The file-based approach is an alternative for applications that do not require functionalities of databases such as consistency of data or transactions. Context log files store every new RDF model collected from context sources on a regular basis (also configured by developers). Afterwards, these files are merged into the persistent context file. Both the triples representation of context and the content of ontologies are stored on the context file. Otherwise, when storing context on databases, ontologies are stored on separate files and read when necessary only (e.g. by the context inference service).

5 Evaluation with an Application on the Educational Domain

TIDIA-Ae is a project aiming at developing and deploying an e-learning infrastructure that exploits a large area high speed Internet network. The basic conceptual model of the project is a core *State* class which may contain recursively other *State* classes and: (a) the *State* class has associations with the *User* class; (b) a *User* may have varied *Roles* (instances of the *Role* class) in different *States*; (c) *Tools* are made available to users when in a given *State*; (d) the *State* class has relationships with the *Contents* class — the aim is to allow *Users* to access different *Contents* and *Tools* in a given *State* depending on their *Role*.

As designed, a typical use of the TIDIA-Ae infrastructure is as follows: a *User* authenticates to enter in a *State*; in that *State* the *User* has a pre-defined *Role* (say *Instructor*) which gives access to pre-defined *Contents* and *Tools*. One of the available tools may be a *Collaborative Editor* which allows new *Contents* to be generated. Other available tools may be an *Instant Messenger*, which allows several participants to communicate synchronously, or a *Whiteboard* tool which, running on a large electronic whiteboard or on portable tablet PCs, may be used to deliver a class either in a traditional classroom or laboratory setting (with students and instructor in the same room) or a distributed mode (with participants in different physical locations). Moreover, the information captured by the *Whiteboard* tool can be used to generate new *Contents* for that *State*.

Considering that the Semantic Context Kernel has been designed to support building context-aware applications, we have investigated its use to allow context-dependent operations to be integrated with the TIDIA-Ae infrastructure. Such integration would allow rules and queries supporting services such as:

– An instructor (*actor* with a *Role*) wishes to be notified (via a validated *rule*) when a given number of students (*actor*s with other *Role*s) is engaged in a conversation supported by the *Instant Messenger* (*software platform*) in any of the courses (*Activity*) he is responsible for. Moreover, this may be associated with an inference that, if the students are also viewing the same *Contents* (e.g. from the *document ontology*), they are in a *Study Group Meeting*.
– A participant (instructor or student, an *actor* with a *Role*) wishes to be notified when some new *Contents* is created via the *Collaborative Editor* (*software platform*) or via the use of the *Whiteboard* tool (*software platform*) in any of the courses (*Activity*) he is involved with — the rule may also specify that the notification may occur only when a tablet PC is used as *hardwarePlaftorm*.

From our investigation so far, we have already learned that most of the information we need for this application can be modelled by the following ontologies: actors, time, activities, and devices. However, we have also identified that the location ontology should be extended to support both physical and virtual locations: if a *Location* could be a virtual location, a *State* would be modelled as a virtual location. We have also identified that some of the ontologies we deal with an *Actor* profile, in particular *Document* and *Project*, may be alternatively associated with an *Activity* as well.

Finally, both the context query service and the context inference service would be used to support high-level services.

6 Concluding Remarks

Ubiquitous computing applications must have access to context information in order to be able to adapt their services accordingly to users' needs. We illustrated our vision by means of the Semantic Context Kernel, an ongoing project that provides semantic-enhanced services for the management of context information gathered from ubicomp environments. The main contribution includes semantic services that can be customized following context-aware applications' requirements toward making it easier the task of development of such applications. These semantic services allow applications not only to store and query context, but also to reason about context in a configurable fashion.

The Semantic Context Kernel has been built on top of an ontological context model for ubicomp applications. This context model has borrowed concepts from consensus Semantic Web ontologies due to the amount of information that they describe. Importing of such ontologies would overload the process of inference over context, even though being an in-memory process.

We have also illustrated the use of the Semantic Context Kernel from the perspective of an application on the educational domain, which demonstrates its value at the same time that allows the identification of possible extensions.

Some key points about context have not been considered yet in our work such as inconsistence, freshness, and privacy of context. We assume context sources as accurate and updated context providers. Privacy of context is also a serious issue on context-aware computing [18]. For instance, we should support privacy of location information when context consumers need to track someone's whereabouts.

Regarding the Semantic Context Kernel infrastructure, future work includes a plug-in support to external inference engines so as to increase the configurability of the context inference service. The context query service can also be extended to support SPARQL queries [19], a W3C effort for a standard query language for RDF.

Acknowledgments

Renato Bulcão Neto is PhD candidate supported by FAPEMA (03/345). Maria Pimentel and Cesar Teixeira are supported by FAPESP and CNPq. This work is supported by Hewlett-Packard, in the context of the *Applied Mobile Technology Solutions in Learning Environments Project*, and is funded by FAPESP in the context of the TIDIA-Ae Project (http://tidia-ae.incubadora.fapesp.br).

References

1. Dey, A.K.: Understanding and using context. Personal and Ubiquitous Computing Journal **5** (2001) 4–7
2. Philipose, M., Fishkin, K.P., Perkowitz, M., Patterson, D.J., Fox, D., Kautz, H., Hahnel, D.: Inferring activities from interactions with objects. IEEE Pervasive Computing **3** (2004) 50–57
3. Helal, S.: Programming pervasive spaces. IEEE Pervasive Computing **4** (2005) 84–87
4. Chen, H., Finin, T., Joshi, A.: Semantic Web in the context broker architecture. In: Proceedings of the International Conference on Pervasive Computing and Communications. (2004) 277–286
5. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientifi c American **284** (2001) 35–43
6. Ranganathan, A., Campbell, R.H.: An infrastructure for context-awareness based on fi rst order logic. Personal Ubiquitous Computing **7** (2003) 353–364
7. Gu, T., Pung, H.K., Zhang, D.Q.: A service-oriented middleware for building context-aware services. Journal of Network and Computer Applications **28** (2005) 1–18
8. Bulcão Neto, R.F., Pimentel, M.G.C.: Toward a domain-independent semantic model for context-aware computing. In: Proceedings of the 3rd Latin American Web Congress, IEEE Press (2005) 61–70
9. Abowd, G.D., Mynatt, E.D., Rodden, T.: The human experience. IEEE Pervasive Computing **1** (2002) 48–57
10. Truong, K.N., Abowd, G.D., Brotherton, J.A.: Who, what, when, where, how: Design issues of capture & access applications. In: Proceedings of the International Conference on Ubiquitous Computing. (2001) 209–224
11. Beckett, D.: RDF/XML syntax specifi cation (revised) (2004) http://www.w3.org/TR/rdf-syntax-grammar/.
12. Klyne, G., Carroll, J.J.: Resource Description Framework (RDF): concepts and abstract syntax (2004) http://www.w3.org/TR/rdf-concepts/.
13. Brickley, D., Guha, R.V.: RDF vocabulary description language 1.0: RDF Schema (2004) http://www.w3.org/TR/rdf-schema/.
14. Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A.: OWL Web ontology language reference (2004) http://www.w3.org/TR/owl-ref/.
15. Bulcão Neto, R.F., Pimentel, M.G.C.: Semantic interoperability between context-aware applications. In: Proceedings of the Brazilian Symposium on Multimedia and Web Systems. (2003) 371–385 (In Portuguese).
16. Bulcão Neto, R.F., Jardim, C.O., Camacho-Guerrero, J.A., Pimentel, M.G.C.: A web service approach for providing context information to CSCW applications. In: Proceedings of the 2nd Latin American Web Congress, IEEE Press (2004) 46–53
17. Miller, L., Seaborne, A., Reggiori, A.: Three implementations of SquishQL: a simple RDF query language. In: Proceedings of the International Semantic Web Conference. (2002) 423–435
18. Lahlou, S., Langheinrich, M., Rocker, C.: Privacy and trust issues with invisible computers. Communications of the ACM, Special issue: The disappearing computer **48** (2005) 59–60
19. Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF (2005) http://www.w3.org/TR/rdf-sparql-query/.